1. **Khái niệm tính kế thừa trong OOP, phạm vi truy cập trong kế thừa, cách thức kế thừa 1 class trong java với từ khóa extend, super.**

Khái niệm tính kế thừa.

Kế thừa là sự liên quan giữa hai class với nhau, trong đó có class cha (superclass) và class con (subclass). Khi kế thừa class con được hưởng tất cả các phương thức và thuộc tính của class cha. Tuy nhiên, nó chỉ được truy cập các thành viên **public** và **protected** của class cha. Nó không được phép truy cập đến thành viên **private** của class cha.

Tư tưởng của kế thừa trong java là có thể tạo ra một class mới được xây dựng trên các lớp đang tồn tại. Khi kế thừa từ một lớp đang tồn tại bạn có sử dụng lại các phương thức và thuộc tính của lớp cha, đồng thời có thể khai báo thêm các phương thức và thuộc tính khác.

Từ khóa **extends** được sử dụng để thể hiện sự kế thừa của một lớp.

Phạm vi truy cập trong kế thừa

	public	*protected*	*mặc định*	*private*
Cùng lớp	✓	✓	✓	✓
Lớp bất kỳ cùng gói	✓	✓	✓	✗
Lớp con khác gói	✓	✓	✗	✗
Lớp bất kỳ khác gói	✓	✗	✗	✗

Cách thức kế thừa.

```
class Employee {
    float salary = 40000;
  }
  class Programmer extends Employee {
    int bonus = 10000;
    public void salary(){
        System.out.println("Luong Lap trinh vien la: " + super.salary);
    }
    public void bonus(){
        System.out.println("Bonus cua Lap trinh vien la: " + bonus);
    }
    public static void main(String args[]) {
      Programmer p = new Programmer();
      p.salary();
      p.bonus();
    }
  }
```

super. sẽ truy cập đến biến và phương thức của lớp cha.

super() sẽ gọi contructor của lớp cha.

2. **Thế nào là Override (ghi đè), Overload (nạp chồng). Phân biệt, tìm hiểu phương thức main có thể nạp chồng được không?**

Override

Override là một tính năng cho phép một lớp con cung cấp một triển khai cụ thể của phương thức đã được cung cấp bởi một trong các lớp cha của nó. Nói một cách khác, ghi đè phương thức là nếu lớp con có một hoặc nhiều phương thức giống với một trong các lớp cha của nó.

Overload

Nạp chồng phương thức là có vài phương thức trùng tên nhưng khác nhau về đối số trong cùng 1 lớp. Nạp chồng phương thức cho phép ta tạo nhiều phiên bản của một phương thức, mỗi phiên bản chấp nhận một danh sách đối số khác nhau, nhằm tạo thuận lợi cho việc gọi phương thức.

Phân biệt override và overload

	Override	Overload
Hành vi	Thay đổi hành vi hiện tại của phương thức.	Thêm hoặc mở rộng cho hành vi của phương thức.
Đa hình	Thể hiện tính đa hình tại run time.	Thể hiện tính đa hình tại compile time.
Danh sách tham số	Danh sách tham số phải giống nhau.	Danh sách tham số có thể khác nhau.
Quyền truy cập	Phương thức ghi đè ở lớp con phải có quyền truy cập bằng hoặc lớn hơn phương thức được ghi đè ở lớp cha.	Các phương thức nạp chồng có thể có quyền truy cập khác nhau.
Giá trị trả về	Kiểu trả về bắt buộc phải giống nhau.	Kiểu trả về có thể khác nhau
Phạm vi	Xảy ra giữa 2 class có quan hệ kế thừa	Xảy ra trong phạm vi cùng 1 class.

Phương thức main có nạp chồng được hay không?

Trong Java, chúng ta có thể hoàn toàn nạp chồng phương thức main(), tức là trong một lớp chúng ta có thể có nhiều hàm main() bằng cách nạp chồng phương thức. Nhưng khi biên dịch thì trình biên dịch sẽ chỉ gọi phương thức main() có đối số truyền vào là một mảng các chuỗi ký tự.

3. **Tìm hiểu về ghi đè equals và hashCode, tại sao khi ghi đè equals lại phải ghi đè thêm hashcode.**

Equals

Khi so sánh hai đối tượng với nhau, Java gọi phương thức equals() của chúng trả về true nếu hai đối tượng bằng nhau hoặc false nếu hai đối tượng là khác nhau. Lưu ý rằng phép so sánh sử dụng phương thức equals() so với sử dụng toán tử == là khác nhau.

Đây là sự khác biệt:

Phương thức equals() được thiết kế để so sánh hai đối tượng về mặt ngữ nghĩa (bằng cách so sánh các thành viên dữ liệu của lớp), trong khi toán tử == so sánh hai đối tượng về mặt kỹ thuật (bằng cách so sánh các tham chiếu của chúng, nghĩa là địa chỉ bộ nhớ).

Hashcode

Số băm này được sử dụng bởi các collection dựa trên bảng băm như Hashtable , HashSet và HashMap để lưu trữ các đối tượng trong các container nhỏ được gọi là "nhóm". Mỗi nhóm được liên kết với mã băm và mỗi nhóm chỉ chứa các đối tượng có mã băm giống hệt nhau.

Nói cách khác, một bảng băm nhóm các phần tử của nó bằng các giá trị mã băm của chúng. Sự sắp xếp này giúp cho bảng băm định vị một phần tử một cách nhanh chóng và hiệu quả bằng cách tìm kiếm trên các phần nhỏ của collection thay vì toàn bộ collection.

Tại sao ghi đè equals lại phải ghi đè hashCode

Khi chỉ ghi đè phương thức equals mà không ghi đè hashCode thì phương thức hashCode sẽ được thừa kế từ lớp Object trả về các địa chỉ bộ nhớ của mỗi đối tượng không nhất quán với phương thức equals, vi phạm sự tương quan giữa equals và hashCode. Vì vậy, khi ghi đè equals thì phải ghi đè hashCode.

4. Java giải quyết vấn đề đa kế thừa như nào? Tìm hiểu sự khác nhau giữa abstract class vs interface

- Vấn đề đa kế thừa => dung inteface
- Một abstract class là một lớp mà không thể được khởi tạo trực tiếp. Nó được sử dụng để tạo ra các lớp con khác. Abstract class có thể chứa cả các phương thức được triển khai và các phương thức trừu tượng. Một lớp con kế thừa từ abstract class phải triển khai (override) tất cả các phương thức trừu tượng trong abstract class đó.

Một interface là một tập hợp các phương thức trừu tượng mà một lớp có thể triển khai. Một interface không thể có các phương thức được triển khai. Nó chỉ định các phương thức mà một lớp triển khai phải cung cấp. Một lớp có thể triển khai nhiều interface, cho phép nó đạt được đa kế thừa thông qua interface. Lớp triển khai một interface phải triển khai tất cả các phương thức trong interface đó.

Một số điểm khác nhau giữa abstract class và interface trong Java bao gồm:

1.1. Triển khai: Một lớp có thể triển khai nhiều interface, nhưng chỉ kế thừa từ một abstract class.

2. Trạng thái: Abstract class có thể có trạng thái (state) bên trong nó, bao gồm các trường (fields) và các biến thành viên (member variables), trong khi interface không thể có trạng thái.

3. Xử lý các phương thức: Abstract class có thể triển khai các phương thức, trong khi interface chỉ định các phương thức mà lớp triển khai phải cung cấp.

4. Đa kế thừa: Một lớp không thể kế thừa từ nhiều abstract class, nhưng có thể triển khai nhiều interface.

5. Upcasting và downcasting, Chuyển đổi (cast) giá trị của từng kiểu dữ liệu

- Upcasting là quá trình chuyển đổi từ một kiểu dữ liệu con sang kiểu dữ liệu cha tương ứng. Nó được thực hiện ngầm định và không cần dùng từ khóa cast. Trong upcasting, không có mất mát dữ liệu và không có rủi ro về kiểu. Ví dụ:

```
class Animal { }
class Dog extends Animal { }
public class Main {
    public static void main(String[] args) {
            Dog dog = new Dog();
            Animal animal = dog; // Upcasting
    }
}
```

- Downcasting là quá trình chuyển đổi từ một kiểu dữ liệu cha sang kiểu dữ liệu con tương ứng. Nó phải được thực hiện một cách rõ ràng bằng cách sử dụng từ khóa cast. Trong downcasting, có mất mát dữ liệu và có rủi ro về kiểu. Do đó, trước khi thực hiện downcasting, cần kiểm tra xem đối tượng có thể được chuyển đổi sang kiểu con không bằng cách sử dụng toán tử instanceof. Ví dụ:

```
class Animal { }
class Dog extends Animal { }
public class Main {
    public static void main(String[] args) {
            Animal animal = new Dog();
            if (animal instanceof Dog) {
                    Dog dog = (Dog) animal; // Downcasting
                    // Sử dụng dog như một đối tượng kiểu Dog
            }
    }
}
```

6. Các lớp Wrapper (Interger, Double, Long, Float,...), Wrapper class dùng để làm gì?

- Wrapper class được sử dụng để đóng gói giá trị nguyên thủy thành đối tượng, hỗ trợ kiểm tra và xử lý dữ liệu, tương thích với generics và các API, và thao tác với các cấu trúc dữ liệu Collection trong Java.